Laboratorio Nro. 1: Implementación de grafos.

Manuela Valencia Toro
Universidad Eafit
Medellín, Colombia
mvalenciat@eafit.edu.co

Laura Sánchez Córdoba
Universidad Eafit
Medellín, Colombia
lsanchezc@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. El método solución (leerArchivo()) crea un grafo, luego procede a leer el archivo .txt mediante un BufferedReader línea a línea, se parte según los espacios en blanco y lo guarda en un arreglo de strings, cuyos datos se pasan a entero y se guardan en el grafo, como sus arcos y con sus respectivos pesos. Cabe agregar que por eficiencia en el uso de memoria se utilizó la estructura de lista enlazada como una representación de listan de adyacencia. Finalmente, se imprimen instrucciones para dibujar el grafo.

2. Es mas conveniente usar listas de adyacencia cuando se maneja grandes cantidades de información, ya que estas ocupan menos espacio que las matrices de adyacencia, debido a que solo se crean los nodos que se van a utilizar, mientras que las matrices crean nodos que muchas veces no van a ser utilizados. Por otro lado, si se van a manejar pequeñas cantidades de información es mejor usar matrices, ya que pueden ser un poco más sencillas de manejar y entender.

3. En este caso es mejor usar listas de adyacencia, ya que debe almacenarse gran cantidad de información, y de esta manera, puede ocuparse menos espacio en memoria.

4. Es mejor usar listas de adyacencia, ya que la cantidad de información que debe almacenarse es demasiado grande, y con matrices ocuparía muchísima más memoria.

5. En este caso es mejor usar una matriz de adyacencia, ya que va a manejarse una pequeña cantidad de información, y, además, es la menor manera de representar una tabla.

6. Cuando el usuario llama al método recibir(), el cual mediante un scanner se introducen los datos por consola (el número de vértices, arcos y dónde se sitúa cada uno). Con el número de vértices se crea un grafo representado en una matriz (ya que es más sencillo acceder a ella y poner los arcos). Mediante un ciclo, el cual verifica en cuales líneas hay indicada una relación mediante la longitud.

   Se verifica que la matriz pueda colorearse mediante el método booleano isBipartite que recibe la matriz y el primer vértice, genera un arreglo con los vértices y lo llena con -1 en representación de un color, se crea una LinkedList que almacena las posiciones para ver si tienen un color o no, en caso de que no se colorea y verifica que no haya dos vértices consecutivos de igual color. Finalmente se muestra si es o no es posible colorearlo.
7.

```
import java.util.*;
class Bipartite
{
   static int V=-1;
   public static void recibir(){
      String a= ""; //c
      Scanner s = new Scanner (System.in); //c
      int matriz [] []; //c
      V = s.nextInt(); //c
      while(V!=0){ //V
         s = new Scanner (System.in); //c
         int arista=s.nextInt(); //c
         s = new Scanner (System.in); //c
         String conexiones = s.nextLine (); //c
         matriz= new int [V] [V];   //c

         while(conexiones.length()>1){ //E
            matriz[(Integer.parseInt(conexiones.substring(0,1)))][(Integer.parseInt(conexiones.substring(2)))]=1;  //
c
            conexiones = s.nextLine ();   //c
         }
         //c + (V*E)

         if(isBipartite(matriz,0))   //(E*V), del método
            a=a + "BICOLORABLE \n"; //c
            else
            a=a + "NOT BICOLORABLE \n"; //c
         V=Integer.parseInt(conexiones); //c
      }
      System.out.println(a);
   }
    T(V,E)= c+(V*(E+E*V))
    T(V,E)= (V*(E+E*V)) R.S.
    T(V,E)= (E+E*V) R.P.
    T(V,E)= (E*V) R.S.
    O(V,E)= (E*V)

   static boolean isBipartite(int G[][],int src)
   {
      int colorArr[] = new int[V]; //c
      for (int i=0; i<V; ++i) //V
         colorArr[i] = -1;   //c

      colorArr[src] = 1; //c
      LinkedList<Integer>q = new LinkedList<Integer>(); //c
      q.add(src); //O(1)

      while (q.size() != 0) //E
      {
         int u = q.poll();  //O(1)
```

```
        if (G[u][u] == 1) //c
            return false;

        for (int v=0; v<V; ++v)//V
        {

            if (G[u][v]==1 && colorArr[v]==-1)//c
            {
                colorArr[v] = 1-colorArr[u];  //c
                q.add(v);//c
            }

            else if (G[u][v]==1 && colorArr[v]==colorArr[u])//c
                return false;  //c
        }
    }
        return true;//c
    }
}
```

T(V,E)=(c+V+(E*V))
T(V,E)=(V+(E*V)) R.S.
T(V,E)=(E*V) R.S.
O(V,E)=(E*V)

8. En la complejidad la variable E determina la cantidad de arcos que tiene el grafo, es decir las relaciones existentes entre los vértices y V es el número de vértices del grafo, es decir, en el peor de los casos cada vértice tiene igual cantidad de arcos.

4) Simulacro de Parcial

1.

	0	1	2	3	4	5	6	7
0				1	1			
1	1		1			1		
2					1		1	
3								1
4			1					
5								
6			1					
7								

2.

0 → 3,4
1 →0,2,5
2 → 4, 6
3 → 7
4 →2
5 →
6 →2
7 →

3,4
0,2,5
4, 6
7
2

6 →2
7 →

CIO TORO BERMÚDEZ
00 Ext. 9473. Oficina: 19 - 627
obe@eafit.edu.co

3. b

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co